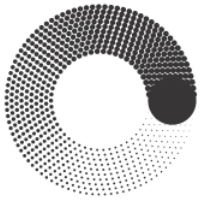
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ РАБОТА) № 3

Дисциплина: Шаблоны проектирования

Выполнил(а): студент(ка) группы 221-371

Коваленко Владислав Артемович
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

Проверил: _____ _____
(Фамилия И.О., степень, звание) (Оценка)

Дата, подпись _____ _____
(Дата) (Подпись)

Москва
2024

Задание

Цель: Разработать систему кэширования для хранения и извлечения часто используемых данных.

Описание: Кэширование предполагает временное хранение часто запрашиваемых данных для более быстрого будущего доступа, что уменьшает потребность в трудоемких операциях. Для создания этой системы используется паттерн Proxy.

Используемый паттерн

Паттерн Proxy

Паттерн Proxy используется для создания заместителя или суррогата другого объекта. В этом проекте Proxy используется для перехвата запросов к основному источнику данных. Он сначала проверяет наличие запрашиваемых данных в кэше и, если данные найдены, возвращает их из кэша. В противном случае он получает данные из основного источника, сохраняет их в кэше и возвращает их пользователю. Это помогает уменьшить количество обращений к основному источнику данных, улучшая производительность и снижая задержки.

Программный код

DataSource.cs
csharp
Копировать код

```
using System;
using System.Collections.Generic;

public class DataSource
{
    private List<string> _data;

    public DataSource()
    {
        // Инициализируем основной источник данных
        _data = new List<string> { "Data1", "Data2", "Data3" };
    }

    public string GetData(int id)
    {
        // Имитация задержки при доступе к данным
        System.Threading.Thread.Sleep(1000);
        return _data[id];
    }
}
```

Cache.cs
csharp
Копировать код

```
using System;
using System.Collections.Generic;

public class Cache
{
```

```
    private Dictionary<int, string> _cache;
    private TimeSpan _cacheDuration;
    private Dictionary<int, DateTime> _cacheExpiry;

    public Cache(TimeSpan cacheDuration)
    {
        _cache = new Dictionary<int, string>();
        _cacheDuration = cacheDuration;
        _cacheExpiry = new Dictionary<int, DateTime>();
    }

    public bool TryGetValue(int key, out string value)
    {
        if (_cache.ContainsKey(key) && _cacheExpiry[key] > DateTime.Now)
        {
            value = _cache[key];
            return true;
        }
        else
        {
            value = null;
            return false;
        }
    }

    public void Add(int key, string value)
    {
        _cache[key] = value;
        _cacheExpiry[key] = DateTime.Now.Add(_cacheDuration);
    }

    public void Remove(int key)
    {
        if (_cache.ContainsKey(key))
        {
            _cache.Remove(key);
            _cacheExpiry.Remove(key);
        }
    }
}
```

DataProxy.cs

csharp

Копировать код

```
using System;

public class DataProxy
{
    private DataSource _dataSource;
    private Cache _cache;

    public DataProxy(DataSource dataSource, Cache cache)
    {
```

```
        _dataSource = dataSource;
        _cache = cache;
    }

    public string GetData(int id)
    {
        if (_cache.TryGetValue(id, out var cachedData))
        {
            Console.WriteLine("Fetching data from cache...");
            return cachedData;
        }
        else
        {
            Console.WriteLine("Fetching data from source...");
            var data = _dataSource.GetData(id);
            _cache.Add(id, data);
            return data;
        }
    }
}
```

Program.cs

csharp

Копировать код

```
using System;

class Program
{
    static void Main()
    {
        var dataSource = new DataSource();
        var cacheDuration = TimeSpan.FromSeconds(10); // Установите время жизни кэша
        var cache = new Cache(cacheDuration);
        var proxy = new DataProxy(dataSource, cache);

        // Запрос данных несколько раз для тестирования
        for (int i = 0; i < 5; i++)
        {
            Console.WriteLine(proxy.GetData(0));
            System.Threading.Thread.Sleep(2000); // Имитация задержки между запросами
        }
    }
}
```